FORSCHUNGSZENTRUM JÜLICH GmbH
Zentralinstitut für Angewandte Mathematik
D-52425 Jülich, Tel. (02461) 61-6402

Interner Bericht

A Parallel Version of the Unsymmetric Lanczos Algorithm and its Application to QMR

H. Martin Bücker, Manfred Sauren*

KFA-ZAM-IB-9605

März 1996
(Stand 01.03.96)

(*) The research of this author was supported by the Graduiertenkolleg “Informatik und Technik”, RWTH Aachen, D–52056 Aachen, Germany.

A Parallel Version of the Unsymmetric Lanczos Algorithm and its Application to QMR

H. Martin Bücker* and Manfred Sauren
{m.buecker,m.sauren}@kfa-juelich.de

Zentralinstitut für Angewandte Mathematik
Forschungszentrum Jülich GmbH (KFA), D–52425 Jülich, Germany

March 1, 1996

Abstract

A new version of the unsymmetric Lanczos algorithm without look-ahead is described combining elements of numerical stability and parallel algorithm design. Firstly, stability is obtained by a coupled two-term procedure that generates Lanczos vectors scaled to unit length. Secondly, the algorithm is derived by making all inner products of a single iteration step independent such that global synchronization on parallel distributed memory computers is reduced. Among the algorithms using the Lanczos process as a major component, the quasi-minimal residual (QMR) method for the solution of systems of linear equations is illustrated by an elegant derivation. The resulting QMR algorithm maintains the favorable properties of the Lanczos algorithm while not increasing computational costs as compared with its corresponding original version.

1 Introduction

The key to many algorithms in engineering and scientific applications often involves the solution of large-scale eigenvalue problems that model oscillations or stability of some physical situation. Such problems are crucial in the analysis of complex dynamic systems. For example, computational physicists yield information about energy levels by investigating eigenvalue spectra in quantum mechanics. Structural engineers are led to eigenvalue problems in vibration analysis where the dynamic equations of motion model a structure subjected to a time-varying force. Control engineers use eigenanalysis to study the stability of large electric power systems whose response to small disturbances around an operating state is represented by a set of linearized equations. This list could easily be extended to molecular dynamics, plasma physics and others; see [27, 8] for illustrations from science and engineering.

Due to their overwhelming importance in applications, eigenproblems are among the most investigated issues in numerical linear algebra. Standard serial algorithms

*The research of this author was supported by the Graduiertenkolleg “Informatik und Technik”, RWTH Aachen, D–52056 Aachen, Germany.

for computing eigenvalues or eigenvectors of a dense matrix typically include two phases. The matrix is initially reduced to a condensed form, e.g. upper Hessenberg form, and afterwards an iterative technique, e.g. QR iteration, is applied to the condensed form [21]. The idea behind this two-phase structure is that the computational complexity of the iterative technique is tremendously decreased when applied to a condensed form rather than to the original matrix. The two-phase structure originally designed for serial computers is easily transferred to parallel shared memory and vector computers commonly used for large problems. For such computer systems, software libraries [1, 15] provide important computational components for the first phase, the reduction of dense matrices to condensed forms. Distributed memory computers are considered in a recent software project [6, 15]. Unfortunately, the parallelization of the second phase is much harder and has initiated the design of new algorithms that are not parallel versions of serial ones; see Section 6.5 of [14].

While the situation is solved quite satisfactorily for dense matrices of moderate size, the setting is different for larger matrices where the above process becomes more and more computationally intractable. However, many problems arising in science and engineering lead to large *sparse* matrices, i.e., the matrix contains enough zero entries to be worth taking advantage of them to reduce both storage requirement and computational work. For large sparse matrices, the Lanczos algorithm comes into the picture. In 1950, Lanczos [29] proposed an iterative method that computes the eigenvalues of a not necessarily symmetric matrix by reducing the matrix via similarity transformation to tridiagonal form. Therefore, the eigenvalues of the two matrices coincide and applying any efficient and numerically stable method to the tridiagonal matrix yields the eigenvalues of the original matrix. This process is particularly suited for large sparse matrices because a matrix need not be stored explicitly; the Lanczos algorithm solely makes use of a matrix, say $\mathbf{A}$, in the form of computing $\mathbf{A}\mathbf{x}$ and $\mathbf{A}^T\mathbf{x}$ for an arbitrary vector $\mathbf{x}$. This feature makes the Lanczos algorithm attractive for large problems where storage usually is the limiting factor. Some historical remarks concerning the Lanczos algorithm are given in [22].

The Lanczos algorithm for symmetric matrices has been analyzed extensively in the literature [9, 31] but has gained much less attention in the unsymmetric case. The reason is that the unsymmetric Lanczos algorithm suffers from numerical instabilities and its potential to fail for unsuitable starting vectors. More recently, several adjustments referred to as *look-ahead* techniques that prevent the process to break down have been proposed. Rather than discussing such techniques that can be found in [33, 24, 25, 18, 32, 4, 36, 26] we consider the introduction of parallelism into the unsymmetric Lanczos algorithm in its look-ahead-free form.

It has been observed [10, 11, 12, 23, 5] that, among the operations involved in Lanczos-like procedures, the bottleneck on distributed memory parallel computers is often due to the computation of inner products. An intuitive argument is as follows. Assume that all vectors are split equally among the processors. Then, inner products require *global* communication, i.e., communication of all processors. On the other hand, all remaining operations do not require any or at least much less communication. For example, on many occasions, the sparsity of the matrix leads to the exchange of messages only between nearby processors when computing $\mathbf{A}\mathbf{x}$ and $\mathbf{A}^T\mathbf{x}$. Consequently, if the number of processors becomes large the performance is dramatically influenced by the computation of inner products. Moreover, inner products possibly

provoke global synchronization points. A *global synchronization point* is defined as the locus of an algorithm at which all local information have to be globally available in order to continue the computation.

Two different strategies have basically been pursued to remedy the difficulties with the parallel computation of inner products. The first is to restructure a serial algorithm such that communication can be overlapped by useful computation, a feature that most distributed memory parallel computers support. The second approach is to design new algorithms with a reduced number of global synchronization points. The pros and cons of these two strategies are added up in [14] for the conjugate gradient method, an iterative technique for the solution of systems of linear equations resembling the Lanczos algorithm in parallelization aspects. Note that following the second approach and deriving a version of the conjugate gradient method that eliminates one global synchronization point requires additional computation [34].

In this paper, we do not consider overlapping but are concerned with the design of a new version of the unsymmetric Lanczos algorithm that has fewer global synchronization points than its corresponding original version. Rather than using a polynomial formulation of the unsymmetric Lanczos algorithm [24, 25] we use an equivalent matrix and vector representation here, and show that the properties of the resulting Lanczos algorithm are transferred to a specific iterative technique for the solution of systems of linear equations called *quasi-minimal residual* (QMR) method [19]. With respect to QMR, we do not only present a new Lanczos algorithm at its heart but additionally introduce a new derivation avoiding the puzzling use of Givens rotations employed in the original paper [19]. We do not address preconditioning techniques here; but note that there has been a true revival of polynomial preconditioning [34] with the availability of parallel computers because this technique decreases the number of inner products, however, at the expense of additional matrix-vector products.

The organization of the paper is as follows. Section 2 reviews the classical unsymmetric Lanczos algorithm. Due to the structure of the underlying equations, this process is referred to as based on *three-term recurrences*. Although most of the work around unsymmetric Lanczos algorithms is influenced by three-term recurrences, it is known that a mathematical equivalent formulation built on *coupled two-term recurrences* is numerically more stable. Section 3 is devoted to the derivation of a new algorithm of that kind additionally improving numerical stability by appropriate vector scalings. The fact that the resulting algorithm has only a single global synchronization point per iteration is essential in the field of parallel computing. This algorithm sets the stage for a new version of QMR derived in the next two subsequent sections. Section 4 proves a lemma that—compared to the original derivation of QMR [19, 20]—considerably eases the entry to the quasi-minimal residual method. Finally, a new QMR version with a single global synchronization point per iteration is derived in Section 5.

Throughout the paper, matrices are denoted by upper case bold letters, vectors are indicated by lower case bold letters, and scalar elements are written in Greek letters. Unless otherwise stated, all matrices belong to the vector space $\mathbb{C}^{N\times N}$, and all vectors are taken from $\mathbb{C}^N$ correspondingly. The vector norm $\|\mathbf{x}\|$ always is the Euclidean norm. The symbol $\mathbf{I} = \mathbf{diag}(1, 1, \ldots, 1)$ represents the identity matrix. A bar denotes complex conjugation of scalars as well as component-wise complex conjugation of vectors, i.e., if $\mathbf{x}^T = (\alpha_1, \alpha_2, \ldots, \alpha_N)$ then $\overline{\mathbf{x}}^T = (\overline{\alpha}_1, \overline{\alpha}_2, \ldots, \overline{\alpha}_N)$. The conjugate transpose of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}^H$.

2 Classical unsymmetric Lanczos algorithm

The classical unsymmetric Lanczos algorithm [29] reduces a matrix $\mathbf{A} \in \mathbb{C}^{N\times N}$ to tridiagonal form $\mathbf{T}$ by using a similarity transformation

$$\mathbf{T} = \mathbf{V}^{-1}\mathbf{A}\mathbf{V} \ . \tag{1}$$

For a moment, assume that $\mathbf{W}$ is used instead of $\mathbf{V}^{-T}$. Then, (1) leads to the following three relations that serve to derive the unsymmetric Lanczos algorithm:

$$\mathbf{W}^T\mathbf{V} = \mathbf{I} \tag{2}$$

$$\mathbf{A}\mathbf{V} = \mathbf{V}\mathbf{T} \tag{3}$$

$$\mathbf{A}^T\mathbf{W} = \mathbf{W}\mathbf{T}^T \ . \tag{4}$$

The algorithm consists of gradually computing the matrices $\mathbf{V}$, $\mathbf{W}$ and $\mathbf{T}$. More precisely, the Lanczos algorithm iteratively computes the column vectors $\mathbf{v}_n$ and $\mathbf{w}_n$ of

$$\mathbf{V} := [\mathbf{v}_1\ \mathbf{v}_2\ \cdots\ \mathbf{v}_N] \quad \text{and} \quad \mathbf{W} := [\mathbf{w}_1\ \mathbf{w}_2\ \cdots\ \mathbf{w}_N]$$

as well as the scalar elements α_n, β_n and γ_n of the tridiagonal matrix

$$\mathbf{T} := \begin{pmatrix} \alpha_1 & \beta_2 & & & & \\ \gamma_2 & \alpha_2 & \beta_3 & & & \\ & \gamma_3 & & & & \\ & & \ddots & \ddots & \ddots & \\ & & & & & \beta_N \\ & & & & \gamma_N & \alpha_N \end{pmatrix} \ .$$

Assume that, in addition to the matrix $\mathbf{A}$, two starting vectors $\mathbf{v}_1$ and $\mathbf{w}_1$ satisfying $\mathbf{w}_1^T\mathbf{v}_1 = 1$ are given. Under the above scenario, the unsymmetric Lanczos algorithm generates the matrix $\mathbf{T}$ and two sequences of vectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \ldots$ and $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \ldots$ called *Lanczos vectors*. Due to (2), the algorithm enforces *biorthogonality* of the Lanczos vectors, i.e.,

$$\mathbf{w}_i^T\mathbf{v}_j = \begin{cases} 0 & \text{if } i \neq j \ , \\ 1 & \text{if } i = j \ . \end{cases} \tag{5}$$

For a derivation of an iterative scheme to compute these vectors, compare the nth columns of (3) and (4)

$$\mathbf{A}\mathbf{v}_n = \gamma_{n+1}\mathbf{v}_{n+1} + \ \alpha_n\mathbf{v}_n + \beta_n\mathbf{v}_{n-1} \ , \tag{6}$$

$$\mathbf{A}^T\mathbf{w}_n = \beta_{n+1}\mathbf{w}_{n+1} + \alpha_n\mathbf{w}_n + \gamma_n\mathbf{w}_{n-1} \ , \tag{7}$$

where $\mathbf{v}_0 = \mathbf{w}_0 = \mathbf{0}$. Solved for $\mathbf{v}_{n+1}$ and $\mathbf{w}_{n+1}$, these equations represent recurrences for the computation of the Lanczos vectors. The coefficients α_n, β_{n+1} and γ_{n+1} are chosen such that the biorthogonality relation (5) holds. In particular, an expression for α_n can be derived alternatively either through multiplication of (6) by $\mathbf{w}_n^T$ or (7) by $\mathbf{v}_n^T$ yielding in both cases

$$\alpha_n = \mathbf{w}_n^T\mathbf{A}\mathbf{v}_n \ . \tag{8}$$

Prior to representing β_{n+1} and γ_{n+1}, set

$$\widetilde{\mathbf{v}}_{n+1} := \mathbf{A}\,\mathbf{v}_n - \alpha_n \mathbf{v}_n - \beta_n \mathbf{v}_{n-1} \ , \tag{9}$$

and

$$\widetilde{\mathbf{w}}_{n+1} := \mathbf{A}^T \mathbf{w}_n - \alpha_n \mathbf{w}_n - \gamma_n \mathbf{w}_{n-1} \ , \tag{10}$$

where β_n and γ_n are assumed to be known from the previous iteration step. Note that from (6) and (7) these settings give

$$\widetilde{\mathbf{v}}_{n+1} = \gamma_{n+1} \mathbf{v}_{n+1} \ , \tag{11}$$

and

$$\widetilde{\mathbf{w}}_{n+1} = \beta_{n+1} \mathbf{w}_{n+1} \ . \tag{12}$$

Hence, the biorthogonality of the Lanczos vectors $\mathbf{v}_{n+1}$ and $\mathbf{w}_{n+1}$ implies the choice of γ_{n+1} and β_{n+1} such that

$$\gamma_{n+1}\beta_{n+1} = \widetilde{\mathbf{w}}_{n+1}^T \widetilde{\mathbf{v}}_{n+1} \ . \tag{13}$$

Properly sequencing the above formulae results in an unsymmetric Lanczos algorithm that—due to the structure of (9) and (10)—is based on *three-term recurrences*. This process is depicted in ALGORITHM 1 where statements leading to global synchronization on distributed memory parallel computers are labeled with a bullet.

Input $\mathbf{A}$, $\mathbf{v}_1$ and $\mathbf{w}_1$ satisfying $\mathbf{w}_1^T \mathbf{v}_1 = 1$
$\beta_1 = \gamma_1 = 0$
$\mathbf{v}_0 = \mathbf{w}_0 = \mathbf{0}$
for $n = 1, 2, 3, \ldots$ **do**

•	$\alpha_n = \mathbf{w}_n^T \mathbf{A}\,\mathbf{v}_n$	Eq. (8)
	$\widetilde{\mathbf{v}}_{n+1} = \mathbf{A}\,\mathbf{v}_n - \alpha_n \mathbf{v}_n - \beta_n \mathbf{v}_{n-1}$	Eq. (9)
	$\widetilde{\mathbf{w}}_{n+1} = \mathbf{A}^T \mathbf{w}_n - \alpha_n \mathbf{w}_n - \gamma_n \mathbf{w}_{n-1}$	Eq. (10)
•	Choose γ_{n+1} and β_{n+1} such that $\gamma_{n+1}\beta_{n+1} = \widetilde{\mathbf{w}}_{n+1}^T \widetilde{\mathbf{v}}_{n+1}$	Eq. (13)
	$\mathbf{v}_{n+1} = \frac{1}{\gamma_{n+1}} \widetilde{\mathbf{v}}_{n+1}$	Eq. (11)
	$\mathbf{w}_{n+1} = \frac{1}{\beta_{n+1}} \widetilde{\mathbf{w}}_{n+1}$	Eq. (12)

endfor

ALGORITHM 1: Three-term procedure with *two* global synchronization points

The coefficients γ_{n+1} and β_{n+1} can be used to scale the Lanczos vectors $\mathbf{v}_{n+1}$ and $\mathbf{w}_{n+1}$, respectively. There are infinitely many choices satisfying (13); three of them are discussed below. A canonical choice used by several authors [21, 27, 13] is to scale one of the sequences of Lanczos vectors, say $\mathbf{v}_{n+1}$, to unit length by setting

$$\gamma_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\| \quad \text{and} \quad \beta_{n+1} = \frac{1}{\gamma_{n+1}} \widetilde{\mathbf{w}}_{n+1}^T \widetilde{\mathbf{v}}_{n+1} \ . \tag{14}$$

Inspired by treating both sequences equally, other scientists [2, 28, 33] propose fixing the scaling parameters as

$$\gamma_{n+1} = \sqrt{|\widetilde{\mathbf{w}}_{n+1}^T \widetilde{\mathbf{v}}_{n+1}|} \quad \text{and} \quad \beta_{n+1} = \frac{1}{\gamma_{n+1}} \widetilde{\mathbf{w}}_{n+1}^T \widetilde{\mathbf{v}}_{n+1} \ . \tag{15}$$

A third choice is motivated by the desire to make the resulting tridiagonal matrix symmetric. This is achieved by writing

$$\gamma_{n+1} = \beta_{n+1} = \sqrt{\widetilde{\mathbf{w}}_{n+1}^T \widetilde{\mathbf{v}}_{n+1}} \ . \tag{16}$$

Restricting the field from $\mathbb{C}$ to $\mathbb{R}$, i.e., given a real matrix and two real starting vectors, the choices (14) and (15) guarantee real arithmetic throughout the whole iteration process, whereas the choice (16) does not preserve real arithmetic. For this latter choice, the resulting tridiagonal matrix is symmetric but generally complex. In contrast to this penalty, choice (16) may improve numerical stability because of its symmetrization of error propagation. Moreover, it may be advantageous to compute the eigenvalues of $\mathbf{T}$, and hence the eigenvalues of $\mathbf{A}$, with an algorithm specifically-designed for symmetric tridiagonal matrices. The reader is referred to [7] for details of the choice (16).

From the numerical analyst's point of view [35, 33, 17, 18], one should in practice scale both sequences of Lanczos vectors to unit length,

$$\|\mathbf{v}_{n+1}\| = \|\mathbf{w}_{n+1}\| = 1, \qquad n = 0, 1, 2, \ldots$$

in order to avoid over- or underflow. This scaling obviously is achievable by setting $\gamma_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\|$ and $\beta_{n+1} = \|\widetilde{\mathbf{w}}_{n+1}\|$. However, (13) solely gives the possibility to choose one of the two coefficients, either γ_{n+1} or β_{n+1}, but not both. To circumvent this difficulty, a second degree of freedom is introduced by bringing into play a nonsingular diagonal matrix $\mathbf{D}$. Assuming that the first degree of freedom is used to scale the Lanczos vectors $\mathbf{v}_n$, the matrix $\mathbf{D}$ can serve for the scaling of the other Lanczos vectors $\mathbf{w}_n$ by setting $\mathbf{W} := \mathbf{V}^{-T}\mathbf{D}$. With this modification, the whole process of deriving an algorithm can be repeated by returning to the starting-point of the discussion: the similarity transformation. We will not call the reader's attention to this process completely. But the first step, the replacement of the similarity transformation (1) by three separate equations analogous to (2)–(4), is given because it forms the basis of the next section.

The modification $\mathbf{W} = \mathbf{V}^{-T}\mathbf{D}$ obviously leads to a different formulation of the biorthogonality (2), but it does not alter (3). Equation (4) slightly changes as can be seen as follows. An equivalent form of (1) is given by

$$\mathbf{T}\mathbf{V}^{-1} = \mathbf{V}^{-1}\mathbf{A} \ .$$

On transposing and applying the modification, the equation

$$\mathbf{W}\mathbf{D}^{-1}\mathbf{T}^T = \mathbf{A}^T\mathbf{W}\mathbf{D}^{-1}$$

follows. Hence in summary, the following analog of (2)–(4)

$$\mathbf{W}^T\mathbf{V} = \mathbf{D} \tag{17}$$

$$\mathbf{A}\mathbf{V} = \mathbf{V}\mathbf{T} \tag{18}$$

$$\mathbf{A}^T\mathbf{W} = \mathbf{W}\mathbf{D}^{-1}\mathbf{T}^T\mathbf{D} \tag{19}$$

can be used to derive a three-term procedure of the unsymmetric Lanczos algorithm where both sequences of Lanczos vectors $\mathbf{v}_n$ and $\mathbf{w}_n$ are scaled to unit length; see Section 2.2 of [20]. Before using these equations in the next section as the starting point of a new version of the unsymmetric Lanczos algorithm we finish with some remarks.

Remark 2.1. In exact arithmetic, the Lanczos algorithm terminates if for some index $\widetilde{\mathbf{w}}_{n+1}^T \widetilde{\mathbf{v}}_{n+1} = 0$. This may occur in two different situations. Firstly, the process regularly stops when $\widetilde{\mathbf{v}}_{n+1} = 0$ or $\widetilde{\mathbf{w}}_{n+1} = 0$; and secondly, the method fails when neither $\widetilde{\mathbf{v}}_{n+1} = 0$ nor $\widetilde{\mathbf{w}}_{n+1} = 0$. The latter case is referred to as *serious breakdown*. In practice, *near breakdowns*, i.e., $\widetilde{\mathbf{w}}_{n+1}^T \widetilde{\mathbf{v}}_{n+1}$ is nonzero but small in some sense, have to be managed. Throughout this paper, we assume that breakdowns will not take place and refer the reader to several *look-ahead* techniques [33, 18, 32, 4, 36, 26] that allow to skip over those iterations where breakdowns occur. Gutknecht [24, 25] presents a theoretical background that offers insight into why breakdowns occur and how they can be treated.

Remark 2.2. The Lanczos algorithm basically consists of the two recurrences (9) and (10) that coincide under two specific conditions. If $\mathbf{A} = \mathbf{A}^H$ and the starting vectors satisfy $\mathbf{w}_1 = \overline{\mathbf{v}}_1$, the process is called *Hermitian* Lanczos algorithm. Supposing $\mathbf{A} = \mathbf{A}^T$ and started with $\mathbf{w}_1 = \mathbf{v}_1$, the method is known as *complex symmetric* Lanczos algorithm [16]. In both cases, work and storage requirements are roughly halved by using reformulated versions of the Lanczos algorithm that omit one of the recurrences. This is possible if the choice of the scaling parameters satisfies $\beta_n = \gamma_n$.

Remark 2.3. What we have shown so far is that ALGORITHM 1 can be derived from the equations (2)–(4). But it is not immediately apparent that, conversely, the quantities given by this algorithm actually fulfill the requirements (2)–(4), particularly the biorthogonality (2). However in exact arithmetic, it is possible to establish that the Lanczos vectors generated by ALGORITHM 1 achieve the biorthogonality relationship. This is demonstrated by Cullum and Willoughby [7] for the particular choice (16). An analogous proof that shows the biorthogonality of the Lanczos vectors generated by a new version of the unsymmetric Lanczos algorithm proposed in this paper is given in the next section.

Remark 2.4. Concerning aspects of parallelism, ALGORITHM 1 consists of two matrix-vector products per iteration, $\mathbf{A}\mathbf{v}_n$ and $\mathbf{A}^T\mathbf{w}_n$, that usually lead to synchronization of only a few processors as opposed to global synchronization needed for the computation of inner products on parallel computers. The choice (13) as well as the computation (8) involve the evaluation of inner products. Unfortunately, there is a direct data dependence between both, i.e., the value of α_n in (8) only can be evaluated once the inner product $\widetilde{\mathbf{w}}_n^T \widetilde{\mathbf{v}}_n$ has been computed in (13) at the previous iteration step. Thus, calculating both inner products simultaneously is prevented. To eliminate this data dependence, the computation of α_n in (8) is reformulated with $\widetilde{\mathbf{v}}_n$ and $\widetilde{\mathbf{w}}_n$ instead of $\mathbf{v}_n$ and $\mathbf{w}_n$. Hence, the calculation in (11) and (12) can be postponed. The result is a restructured Lanczos procedure with a single global synchronization point per iteration given in ALGORITHM 2. Note that (9) is slightly modified by transforming $\mathbf{A}\mathbf{v}_n$ into $\frac{1}{\gamma_n}\mathbf{A}\widetilde{\mathbf{v}}_n$ in order to prevent the calculation of a third matrix-vector product per iteration.

We further remark that the two matrix-vector products of the unsymmetric Lanczos algorithm are independent so that a parallel implementation of ALGORITHM 2 can execute the matrix-vector products $\mathbf{A}\widetilde{\mathbf{v}}_{n+1}$ and $\mathbf{A}^T\mathbf{w}_n$ simultaneously, right after (9) but before (10). Moreover, if (10) is reformulated using $\frac{1}{\beta_n}\mathbf{A}^T\widetilde{\mathbf{w}}_n$ rather than $\mathbf{A}^T\mathbf{w}_n$, ALGORITHM 2 essentially gives Algorithm 2.3 of [28].

Input $\mathbf{A}$, $\widetilde{\mathbf{v}}_1$ and $\widetilde{\mathbf{w}}_1$ satisfying $\widetilde{\mathbf{w}}_1^T\widetilde{\mathbf{v}}_1 \neq 0$

$\mathbf{v}_0 = \mathbf{w}_0 = \mathbf{0}$

Choose β_1 and γ_1 such that $\beta_1\gamma_1 = \widetilde{\mathbf{w}}_1^T\widetilde{\mathbf{v}}_1$

for $n = 1, 2, 3, \ldots$ **do**

- $\alpha_n = \dfrac{\widetilde{\mathbf{w}}_n^T\mathbf{A}\widetilde{\mathbf{v}}_n}{\beta_n\gamma_n}$ Eq. (8)

  $\mathbf{v}_n = \frac{1}{\gamma_n}\widetilde{\mathbf{v}}_n$ Eq. (11)

  $\mathbf{w}_n = \frac{1}{\beta_n}\widetilde{\mathbf{w}}_n$ Eq. (12)

  $\widetilde{\mathbf{v}}_{n+1} = \frac{1}{\gamma_n}\mathbf{A}\widetilde{\mathbf{v}}_n - \alpha_n\mathbf{v}_n - \beta_n\mathbf{v}_{n-1}$ Eq. (9)

  $\widetilde{\mathbf{w}}_{n+1} = \mathbf{A}^T\mathbf{w}_n - \alpha_n\mathbf{w}_n - \gamma_n\mathbf{w}_{n-1}$ Eq. (10)
- Choose β_{n+1} and γ_{n+1} such that $\beta_{n+1}\gamma_{n+1} = \widetilde{\mathbf{w}}_{n+1}^T\widetilde{\mathbf{v}}_{n+1}$ Eq. (13)

endfor

ALGORITHM 2: Three-term procedure with *one* global synchronization point

Remark 2.5. In some applications, e.g. when the Lanczos algorithm is employed as an underlying process for the solution of linear systems of equations, the Lanczos vectors $\mathbf{v}_n$ and $\mathbf{w}_n$ are not explicitly used. Equations (9) and (10) can be reformulated with $\widetilde{\mathbf{v}}_n$ and $\widetilde{\mathbf{w}}_n$ rather than $\mathbf{v}_n$ and $\mathbf{w}_n$. Hence in this case, (11) and (12) can be eliminated completely.

3 Lanczos algorithm based on coupled two-term recurrences

The standard Lanczos algorithm presented in the last section is based on three-term recurrences for the generation of the Lanczos vectors. In this section, a different formulation of the method built on coupled two-term recurrences is derived. Although Lanczos [30] used a similar technique in the early 1950's, the majority of papers are dealing with the three-term procedure, until, quite recently, Freund et al. [18] reused this idea to improve numerical stability. Additionally, to prevent instabilities they scaled both sequences of Lanczos vectors to unit length, as discussed in the previous section. In their algorithm, each iteration involves three global synchronization points that readily can be reduced to two but not to a single one. This paper proposes a different approach that combines the favorable numerical properties with aspects of parallel algorithm design, specifically the avoidance of global synchronization. The result is a new unsymmetric Lanczos algorithm based on coupled two-term recurrences that generates Lanczos vectors scaled to unit length and has only *one* global synchronization point per iteration.

Throughout the rest of this paper, we assume that the tridiagonal matrix has an LU decomposition

$$\mathbf{T} = \mathbf{L}\mathbf{U} \ , \tag{20}$$

where the factors are of bidiagonal form

$$\mathbf{L} := \begin{pmatrix} \tau_1 & & & \\ \gamma_2 & \tau_2 & & \\ & \ddots & \ddots & \\ & & \gamma_N & \tau_N \end{pmatrix} \quad \text{and} \quad \mathbf{U} := \begin{pmatrix} 1 & \mu_2 & & \\ & 1 & \ddots & \\ & & \ddots & \mu_N \\ & & & 1 \end{pmatrix} \ . \tag{21}$$

As mentioned above, an algorithm where the Lanczos vectors are scaled to unit length can be derived from (17)–(19). But this derivation results in three-term recurrences. Our new approach introduces the settings $\mathbf{P} := \mathbf{V}\mathbf{U}^{-1}$ and $\widetilde{\mathbf{Q}} := \mathbf{W}\mathbf{D}^{-1}\mathbf{U}^T$ that produces coupled two-term recurrences as will be shown in the sequel. Applying these settings as well as (20) to (17)–(19) leads to

$$\mathbf{W}^T\mathbf{V} = \mathbf{D} \tag{22}$$

$$\mathbf{V} = \mathbf{P}\mathbf{U} \tag{23}$$

$$\mathbf{A}^T\mathbf{W} = \widetilde{\mathbf{Q}}\mathbf{L}^T\mathbf{D} \tag{24}$$

$$\mathbf{A}\mathbf{P} = \mathbf{V}\mathbf{L} \tag{25}$$

$$\widetilde{\mathbf{Q}} = \mathbf{W}\mathbf{D}^{-1}\mathbf{U}^T \ . \tag{26}$$

Suppose that the matrices introduced by the above settings have column vectors according to

$$\mathbf{P} := [\mathbf{p}_1 \ \mathbf{p}_2 \ \cdots \ \mathbf{p}_N] \ , \qquad \widetilde{\mathbf{Q}} := [\widetilde{\mathbf{q}}_1 \ \widetilde{\mathbf{q}}_2 \ \cdots \ \widetilde{\mathbf{q}}_N] \ ,$$

and that the diagonal matrix is of the form

$$\mathbf{D} := \mathrm{diag}(\delta_1, \delta_2, \ldots, \delta_N) \ .$$

Then, the nth columns of the four equations (23)–(26) are given by

$$\mathbf{v}_n = \mathbf{p}_n + \mu_n \mathbf{p}_{n-1} \tag{27}$$

$$\mathbf{A}^T\mathbf{w}_n = \tau_n\delta_n\widetilde{\mathbf{q}}_n + \gamma_n\delta_n\widetilde{\mathbf{q}}_{n-1} \tag{28}$$

$$\mathbf{A}\mathbf{p}_n = \gamma_{n+1}\mathbf{v}_{n+1} + \tau_n\mathbf{v}_n \tag{29}$$

$$\widetilde{\mathbf{q}}_n = \frac{\mu_{n+1}}{\delta_{n+1}}\mathbf{w}_{n+1} + \frac{1}{\delta_n}\mathbf{w}_n \ , \tag{30}$$

where $\mathbf{p}_0 = \widetilde{\mathbf{q}}_0 = \mathbf{0}$. The substitution $\mathbf{q}_n := \tau_n\delta_n\widetilde{\mathbf{q}}_n$ is used to obtain an equivalent form of (28) and (30) resulting in

$$\mathbf{q}_n = \mathbf{A}^T\mathbf{w}_n - \frac{\gamma_n\delta_n}{\tau_{n-1}\delta_{n-1}}\mathbf{q}_{n-1} \ ,$$

$$\frac{\tau_n\delta_n\mu_{n+1}}{\delta_{n+1}}\mathbf{w}_{n+1} = \mathbf{q}_n - \tau_n\mathbf{w}_n \ ,$$

where $\mathbf{q}_0 = \mathbf{0}$. To eliminate all δ_i's, these two equations are reformulated using

$$\xi_{n+1} = \frac{\tau_n \delta_n \mu_{n+1}}{\delta_{n+1}} \tag{31}$$

that will be needed later for scaling of the Lanczos vectors. Hence, (27)–(30) lead to the four basic equations from which an algorithm will be derived

$$\mathbf{p}_n = \mathbf{v}_n - \mu_n \mathbf{p}_{n-1} \tag{32}$$

$$\mathbf{q}_n = \mathbf{A}^T \mathbf{w}_n - \frac{\gamma_n \mu_n}{\xi_n} \mathbf{q}_{n-1} \tag{33}$$

$$\gamma_{n+1} \mathbf{v}_{n+1} = \mathbf{A}\mathbf{p}_n - \tau_n \mathbf{v}_n \tag{34}$$

$$\xi_{n+1} \mathbf{w}_{n+1} = \mathbf{q}_n - \tau_n \mathbf{w}_n \ , \tag{35}$$

where $\mathbf{p}_0 = \mathbf{q}_0 = \mathbf{0}$. The structure of these four recurrences that form the heart of the algorithm gives rise to calling the process *coupled two-term* procedure. We still have to show that the updates of the unknown quantities are uniquely determined by these relations. More precisely, assuming that the vectors $\mathbf{p}_{n-1}$, $\mathbf{q}_{n-1}$, $\mathbf{v}_n$ and $\mathbf{w}_n$ as well as the coefficients μ_n, γ_n, ξ_n and τ_n are known from the previous iteration step, their successors have to be determined enforcing the biorthogonality relationship. Under these assumptions, the vectors $\mathbf{p}_n$ and $\mathbf{q}_n$ are updated from (32) and (33). Furthermore, the following quantities are computed

$$\widetilde{\mathbf{v}}_{n+1} := \mathbf{A}\mathbf{p}_n - \tau_n \mathbf{v}_n \ , \tag{36}$$

$$\widetilde{\mathbf{w}}_{n+1} := \mathbf{q}_n - \tau_n \mathbf{w}_n \ , \tag{37}$$

and

$$\gamma_{n+1} := \left\| \widetilde{\mathbf{v}}_{n+1} \right\| \ , \tag{38}$$

$$\xi_{n+1} := \left\| \widetilde{\mathbf{w}}_{n+1} \right\| \ . \tag{39}$$

From (34) and (35), the above settings imply that the vectors

$$\mathbf{v}_{n+1} = \frac{1}{\gamma_{n+1}} \widetilde{\mathbf{v}}_{n+1} \quad \text{and} \quad \mathbf{w}_{n+1} = \frac{1}{\xi_{n+1}} \widetilde{\mathbf{w}}_{n+1} \tag{40}$$

are scaled to unit length. Using the definition

$$\varrho_{n+1} := \widetilde{\mathbf{w}}_{n+1}^T \widetilde{\mathbf{v}}_{n+1} \tag{41}$$

the biorthogonality (22) shows

$$\varrho_{n+1} = \gamma_{n+1} \xi_{n+1} \delta_{n+1} \ . \tag{42}$$

Therefore, (31) solved for μ_{n+1} gives

$$\mu_{n+1} = \frac{\gamma_n \xi_n \varrho_{n+1}}{\gamma_{n+1} \tau_n \varrho_n} \ . \tag{43}$$

Note that all operations leading to global synchronization, i.e., (38), (39) and (41), are independent. Thus, they can be computed at one global synchronization point. It remains to be shown that the computation of the last coefficient τ_{n+1} involves no further global synchronization point. To determine τ_{n+1} we multiply (34) by $\mathbf{w}_{n+1}^T$ and obtain

$$\tau_{n+1} = \frac{\mathbf{w}_{n+1}^T \mathbf{A}\,\mathbf{p}_{n+1}}{\mathbf{w}_{n+1}^T \mathbf{v}_{n+1}} \ . \tag{44}$$

On account of (32), the determination of $\mathbf{p}_{n+1}$ involves the knowledge of μ_{n+1} whose value according to (43) only is available after having calculated the inner products of ϱ_{n+1} and γ_{n+1}. Hence, the inner product $\mathbf{w}_{n+1}^T \mathbf{A}\,\mathbf{p}_{n+1}$ in (44) cannot be evaluated with the others simultaneously. We overcome that difficulty as follows. Putting in $\mathbf{p}_{n+1}$ given by (32) implies

$$\tau_{n+1} = \frac{\mathbf{w}_{n+1}^T \mathbf{A}\,\mathbf{v}_{n+1}}{\mathbf{w}_{n+1}^T \mathbf{v}_{n+1}} - \mu_{n+1} \frac{\mathbf{w}_{n+1}^T \mathbf{A}\,\mathbf{p}_{n}}{\mathbf{w}_{n+1}^T \mathbf{v}_{n+1}} \ .$$

Expanding the first term by $\xi_{n+1}\gamma_{n+1}$, inserting $\mathbf{A}\,\mathbf{p}_n$ given by (34) into the second term, and using biorthogonality (22) results in

$$\tau_{n+1} = \frac{\widetilde{\mathbf{w}}_{n+1}^T \mathbf{A}\,\widetilde{\mathbf{v}}_{n+1}}{\widetilde{\mathbf{w}}_{n+1}^T \widetilde{\mathbf{v}}_{n+1}} - \gamma_{n+1}\mu_{n+1} \ .$$

Setting

$$\varepsilon_{n+1} := \widetilde{\mathbf{w}}_{n+1}^T \mathbf{A}\,\widetilde{\mathbf{v}}_{n+1} \ , \tag{45}$$

we obtain

$$\tau_{n+1} = \frac{\varepsilon_{n+1}}{\varrho_{n+1}} - \gamma_{n+1}\mu_{n+1} \ . \tag{46}$$

Note that in the above derivation the initialization $\mathbf{p}_0 = \mathbf{0}$ leads to $\tau_1 = \varepsilon_1/\varrho_1$, and thus $\mu_1 = 0$. The computation of ε_{n+1} requires a further inner product, but all operations involving global synchronization, i.e., (38), (39), (41) and (45), are independent, and can be calculated simultaneously. The resulting algorithm is formulated in ALGORITHM 3.

Remark 3.1. At first sight, one realizes that neither the Lanczos vectors $\mathbf{v}_n$ nor $\mathbf{w}_n$ appear in ALGORITHM 3. So, it seems that this algorithm does not fulfill one of the intended conditions, namely Lanczos vectors $\mathbf{v}_n$ and $\mathbf{w}_n$ that are scaled to unit length. However, the recurrences of the algorithm involve $\frac{1}{\gamma_n}\widetilde{\mathbf{v}}_n$ and $\frac{1}{\xi_n}\widetilde{\mathbf{w}}_n$ expressing the desired quantities as can be seen from (40). A syntactically different formulation of the same algorithm that explicitly uses the Lanczos vectors can be obtained by inserting (40) into ALGORITHM 3.

Remark 3.2. The values $\delta_n = \mathbf{w}_n^T \mathbf{v}_n$ of the diagonal matrix $\mathbf{D}$ that characterize the biorthogonality relationship can be extracted from (42) by inserting the assignment $\delta_n = \varrho_n/(\gamma_n \xi_n)$.

Input $\mathbf{A}$, $\widetilde{\mathbf{v}}_1$ and $\widetilde{\mathbf{w}}_1$ satisfying $\widetilde{\mathbf{w}}_1^T\widetilde{\mathbf{v}}_1 \neq 0$

$\mathbf{p}_0 = \mathbf{q}_0 = \mathbf{0}$

$\gamma_1 = \|\widetilde{\mathbf{v}}_1\|\,,\ \xi_1 = \|\widetilde{\mathbf{w}}_1\|\,,\ \varrho_1 = \widetilde{\mathbf{w}}_1^T\widetilde{\mathbf{v}}_1\,,\ \varepsilon_1 = \left(A^T\widetilde{\mathbf{w}}_1\right)^T\widetilde{\mathbf{v}}_1\,,\ \mu_1 = 0\,,\ \tau_1 = \dfrac{\varepsilon_1}{\varrho_1}$

for $n = 1, 2, 3, \ldots$ **do**

$\mathbf{p}_n = \frac{1}{\gamma_n}\widetilde{\mathbf{v}}_n - \mu_n\mathbf{p}_{n-1}$ Eq. (32)

$\mathbf{q}_n = \frac{1}{\xi_n}\mathbf{A}^T\widetilde{\mathbf{w}}_n - \frac{\gamma_n\mu_n}{\xi_n}\mathbf{q}_{n-1}$ Eq. (33)

$\widetilde{\mathbf{v}}_{n+1} = \mathbf{A}\mathbf{p}_n - \frac{\tau_n}{\gamma_n}\widetilde{\mathbf{v}}_n$ Eq. (36)

$\widetilde{\mathbf{w}}_{n+1} = \mathbf{q}_n - \frac{\tau_n}{\xi_n}\widetilde{\mathbf{w}}_n$ Eq. (37)

- $\gamma_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\|$ Eq. (38)
- $\xi_{n+1} = \|\widetilde{\mathbf{w}}_{n+1}\|$ Eq. (39)
- $\varrho_{n+1} = \widetilde{\mathbf{w}}_{n+1}^T\widetilde{\mathbf{v}}_{n+1}$ Eq. (41)
- $\varepsilon_{n+1} = \left(\mathbf{A}^T\widetilde{\mathbf{w}}_{n+1}\right)^T\widetilde{\mathbf{v}}_{n+1}$ Eq. (45)

$\mu_{n+1} = \dfrac{\gamma_n\xi_n\varrho_{n+1}}{\gamma_{n+1}\tau_n\varrho_n}$ Eq. (43)

$\tau_{n+1} = \dfrac{\varepsilon_{n+1}}{\varrho_{n+1}} - \gamma_{n+1}\mu_{n+1}$ Eq. (46)

endfor

ALGORITHM 3: Coupled two-term procedure with *one* global synchronization point

Remark 3.3. ALGORITHM 3 involves the computation of two matrix-vector products $\mathbf{A}^T\widetilde{\mathbf{w}}_{n+1}$ and $\mathbf{A}\mathbf{p}_n$. These two matrix-vector products of the unsymmetric Lanczos algorithm are independent such that a parallel implementation can compute the two matrix-vector products simultaneously, e.g. as follows. The order of determining $\widetilde{\mathbf{v}}_{n+1}$ and $\widetilde{\mathbf{w}}_{n+1}$ is reversed. Right after the calculation of $\widetilde{\mathbf{w}}_{n+1}$ the two matrix-vector products $\mathbf{A}^T\widetilde{\mathbf{w}}_{n+1}$ and $\mathbf{A}\mathbf{p}_n$ are computed. With this value of $\mathbf{A}\mathbf{p}_n$, the vector $\widetilde{\mathbf{v}}_{n+1}$ is calculated afterwards.

The above discussion illustrates that (22)–(26), particularly the biorthogonality relationship, are used to derive ALGORITHM 3. We finish this section with a theorem conversely showing that, in exact arithmetic, the vectors generated by ALGORITHM 3 are biorthogonal.

Theorem 3.1. *As long as no breakdown occurs, the* $\widetilde{\mathbf{v}}$*-vectors and* $\widetilde{\mathbf{w}}$*-vectors generated by* ALGORITHM 3 *satisfy*

$$\widetilde{\mathbf{w}}_i^T\widetilde{\mathbf{v}}_j = \begin{cases} 0 & \text{if } i \neq j\ , \\ \varrho_i \neq 0 & \text{if } i = j\ . \end{cases} \tag{47}$$

Proof. We prove by induction on n that (47) is true for $1 \leq i, j \leq n$. The assignment $\varrho_1 = \widetilde{\mathbf{w}}_1^T\widetilde{\mathbf{v}}_1$ in the initialization phase of the algorithm gives the basis of the induction. Note that $\varrho_i \neq 0$ as long as no breakdown occurs. For $n \geq 1$, the assignments labeled with (37) and (33) as well as the insertion of $\mathbf{q}_{n-1} = \widetilde{\mathbf{w}}_n + \frac{\tau_{n-1}}{\xi_{n-1}}\widetilde{\mathbf{w}}_{n-1}$

obtained from (37) imply

$$\widetilde{\mathbf{w}}_{n+1} = \frac{1}{\xi_n}\mathbf{A}^T\widetilde{\mathbf{w}}_n - \frac{\gamma_n\mu_n + \tau_n}{\xi_n}\widetilde{\mathbf{w}}_n - \frac{\gamma_n\mu_n\tau_{n-1}}{\xi_n\xi_{n-1}}\widetilde{\mathbf{w}}_{n-1} \ , \tag{48}$$

where the last term is understood to be non-existing for $n = 1$. An analogous calculation using the assignments labeled with (36) and (32) yields

$$\widetilde{\mathbf{v}}_{i+1} = \frac{1}{\gamma_i}\mathbf{A}\widetilde{\mathbf{v}}_i - \frac{\gamma_i\mu_i + \tau_i}{\gamma_i}\widetilde{\mathbf{v}}_i - \frac{\mu_i\tau_{i-1}}{\gamma_{i-1}}\widetilde{\mathbf{v}}_{i-1} \ , \qquad i = 1, \ldots, n \ .$$

By the induction hypothesis, we conclude from the last equation

$$\widetilde{\mathbf{w}}_n^T\mathbf{A}\widetilde{\mathbf{v}}_i = \begin{cases} 0 & \text{if } i = 1, \ldots, n-2 \ , \\ \varrho_n\gamma_{n-1} & \text{if } i = n-1 \ . \end{cases}$$

Therefore, (48) gives

$$\widetilde{\mathbf{w}}_{n+1}^T\widetilde{\mathbf{v}}_i = \frac{1}{\xi_n}\widetilde{\mathbf{w}}_n^T\mathbf{A}\widetilde{\mathbf{v}}_i - \frac{\gamma_n\mu_n + \tau_n}{\xi_n}\widetilde{\mathbf{w}}_n^T\widetilde{\mathbf{v}}_i - \frac{\gamma_n\mu_n\tau_{n-1}}{\xi_n\xi_{n-1}}\widetilde{\mathbf{w}}_{n-1}^T\widetilde{\mathbf{v}}_i$$

$$= \begin{cases} 0 & \text{if } i = 1, \ldots, n-2 \ , \\ \dfrac{\varrho_n\gamma_{n-1}}{\xi_n} - \dfrac{\gamma_n\mu_n\tau_{n-1}}{\xi_n\xi_{n-1}}\varrho_{n-1} & \text{if } i = n-1 \ , \\ \dfrac{\varepsilon_n}{\xi_n} - \dfrac{\gamma_n\mu_n + \tau_n}{\xi_n}\varrho_n & \text{if } i = n \ , \end{cases}$$

where the induction hypothesis and the assignment of ε_n are used. The assignments of μ_n and τ_n imply the biorthogonality for $i = n-1$ and $i = n$, respectively. Similarly, we have that $\widetilde{\mathbf{w}}_i^T\widetilde{\mathbf{v}}_{n+1} = 0$ for $i = 1, \ldots, n$. The assignment of ϱ_{n+1} completes the proof. □

4 A special least-squares problem

In this section, we present the mathematical background needed to offer an easy and new approach to the QMR algorithm. The following lemma gives an iterative solution of a special least-squares problem.

Lemma 4.1. *For $n \in \mathbb{N}$, let $\mathbf{L}_n \in \mathbb{C}^{(n+1)\times n}$ be a lower bidiagonal matrix of full rank having the form*

$$\mathbf{L}_n = \begin{pmatrix} \tau_1 & & \\ \gamma_2 & \ddots & \\ & \ddots & \tau_n \\ & & \gamma_{n+1} \end{pmatrix} \ .$$

Furthermore, with $\gamma_1 \in \mathbb{C}$ and $\mathbf{e}_1^{(n+1)} = (1, 0, \ldots, 0)^T \in \mathbb{C}^{n+1}$ let $\mathbf{y}_n \in \mathbb{C}^n$ denote the unique solution of the least-squares problem

$$\left\| \gamma_1\,\mathbf{e}_1^{(n+1)} - \mathbf{L}_n\,\mathbf{y}_n \right\| \quad = \quad \min_{\mathbf{y}\in\mathbb{C}^n} \left\| \gamma_1\,\mathbf{e}_1^{(n+1)} - \mathbf{L}_n\,\mathbf{y} \right\| \ .$$

Then, the recursion relation

$$\mathbf{y}_n = (1+\theta_n)\begin{pmatrix}\mathbf{y}_{n-1}\\ 0\end{pmatrix} - \theta_n \begin{pmatrix}\mathbf{y}_{n-2}\\ 0\\ 0\end{pmatrix} + \kappa_n\, \mathbf{e}_n^{(n)} \tag{49}$$

holds, where

$$\mathbf{e}_n^{(n)} = (0,\ldots,0,1)^T \in \mathbb{C}^n \ ,$$

and

$$\theta_n = \frac{|\tau_n|^2\,(1-\lambda_n)}{\lambda_n|\tau_n|^2 + |\gamma_{n+1}|^2} \ , \qquad (n \geq 1) \ , \tag{50}$$

$$\kappa_n = \frac{-\gamma_n\overline{\tau}_n\,\kappa_{n-1}}{\lambda_n|\tau_n|^2 + |\gamma_{n+1}|^2} \ , \qquad (n \geq 1) \ , \qquad \kappa_0 = -1 \ , \tag{51}$$

with

$$\lambda_n = \frac{\lambda_{n-1}|\tau_{n-1}|^2}{\lambda_{n-1}|\tau_{n-1}|^2 + |\gamma_n|^2} \ , \qquad (n \geq 2) \ , \qquad \lambda_1 = 1 \ . \tag{52}$$

Proof. The full rank condition implies that if for some index $\tau_i = 0$ then $\gamma_j \neq 0$ for $j = i+1,\ldots,n+1$, such that the sequences (52), (51), (50), and finally (49) are well-defined. Note that (49) gives κ_n as the last component of $\mathbf{y}_n$. Putting

$$\mathbf{M}_n := \mathbf{L}_n^H\mathbf{L}_n = \begin{pmatrix} |\tau_1|^2+|\gamma_2|^2 & \overline{\gamma}_2\tau_2 & & & & \\ \gamma_2\overline{\tau}_2 & |\tau_2|^2+|\gamma_3|^2 & \overline{\gamma}_3\tau_3 & & & \\ & \gamma_3\overline{\tau}_3 & \ddots & & & \\ & & \ddots & \ddots & & \\ & & & & & \overline{\gamma}_n\tau_n \\ & & & & \gamma_n\overline{\tau}_n & |\tau_n|^2+|\gamma_{n+1}|^2 \end{pmatrix} \in \mathbb{C}^{n\times n} \ ,$$

we show by induction on n that $\mathbf{y}_n$ fulfills the normal equation

$$\mathbf{M}_n\mathbf{y}_n = \mathbf{L}_n^H\mathbf{L}_n\mathbf{y}_n = \mathbf{L}_n^H\gamma_1\mathbf{e}_1^{(n+1)} = \gamma_1\overline{\tau}_1\mathbf{e}_1^{(n)} \ .$$

The cases $n = 1$ and $n = 2$ are obvious. For $n > 2$, equation (49) gives

$$\mathbf{M}_n\mathbf{y}_n = (1+\theta_n)\mathbf{M}_n\begin{pmatrix}\mathbf{y}_{n-1}\\ 0\end{pmatrix} \quad - \quad \theta_n\mathbf{M}_n\begin{pmatrix}\mathbf{y}_{n-2}\\ 0\\ 0\end{pmatrix} \quad + \quad \kappa_n\mathbf{M}_n\mathbf{e}_n^{(n)} \ .$$

With $v_i \in \mathbb{C}$ denoting the last component of $\mathbf{y}_i$, the special structure of $\mathbf{M}_n$ leads to

$$\mathbf{M}_n\mathbf{y}_n = (1+\theta_n)\begin{pmatrix}\mathbf{M}_{n-1}\mathbf{y}_{n-1}\\ \gamma_n\overline{\tau}_n v_{n-1}\end{pmatrix} - \theta_n\begin{pmatrix}\mathbf{M}_{n-2}\mathbf{y}_{n-2}\\ \gamma_{n-1}\overline{\tau}_{n-1}v_{n-2}\\ 0\end{pmatrix} + \kappa_n\begin{pmatrix}\mathbf{0}\\ \overline{\gamma}_n\tau_n\\ |\tau_n|^2+|\gamma_{n+1}|^2\end{pmatrix}.$$

For $i < n$, the induction hypothesis gives $v_i = \kappa_i$ and $\mathbf{M}_i \mathbf{y}_i = \gamma_1 \overline{\tau}_1 \mathbf{e}_1^{(i)}$. Hence,

$$\mathbf{M}_n \mathbf{y}_n = \begin{pmatrix} \gamma_1 \overline{\tau}_1 \mathbf{e}_1^{(n-2)} \\ -\theta_n \gamma_{n-1} \overline{\tau}_{n-1} \kappa_{n-2} + \kappa_n \overline{\gamma}_n \tau_n \\ (1+\theta_n) \gamma_n \overline{\tau}_n \kappa_{n-1} + \kappa_n \left(|\tau_n|^2 + |\gamma_{n+1}|^2 \right) \end{pmatrix} .$$

Using (50)–(52), a simple calculation shows that the last two components of the above vector are equal to zero implying $\mathbf{M}_n \mathbf{y}_n = \gamma_1 \overline{\tau}_1 \mathbf{e}_1^{(n)}$. □

The following corollary is used in the next section.

Corollary 4.1. *For $n \in \mathbb{N}$ and $\mathbf{y}_n,\ \mathbf{g}_n \in \mathbb{C}^n$, the coupled iteration*

$$\mathbf{y}_n = \begin{pmatrix} \mathbf{y}_{n-1} \\ 0 \end{pmatrix} + \mathbf{g}_n \ , \tag{53}$$

$$\mathbf{g}_n = \theta_n \begin{pmatrix} \mathbf{g}_{n-1} \\ 0 \end{pmatrix} + \kappa_n \mathbf{e}_n^{(n)} \ , \tag{54}$$

gives the solution of the least-squares problem of Lemma 4.1, where κ_n and θ_n are supplied by (50)–(52)*.*

Proof. By induction, the coupled iteration is identical to (49). □

5 Quasi-minimal residual method

From the two previous sections, we take the Lanczos process given in ALGORITHM 3 as well as Corollary 4.1 as the two main ingredients to derive a method for the solution of a system of linear equations

$$\mathbf{A}\mathbf{x} = \mathbf{b} \ , \quad \text{where} \quad \mathbf{A} \in \mathbb{C}^{N \times N} \quad \text{and} \quad \mathbf{x}, \mathbf{b} \in \mathbb{C}^N \ . \tag{55}$$

The goal is to draw out a new version of an iterative technique called quasi-minimal residual (QMR) method proposed by Freund and Nachtigal [19]. Given any initial guess $\mathbf{x}_0 \in \mathbb{C}^N$ to the solution of (55), the nth QMR iterate is of the form

$$\mathbf{x}_n = \mathbf{x}_0 + \mathbf{V}_n \mathbf{z}_n \ , \tag{56}$$

where $\mathbf{V}_n \in \mathbb{C}^{N \times n}$ is generated by the unsymmetric Lanczos algorithm, and $\mathbf{z}_n \in \mathbb{C}^n$ is determined by a quasi-minimal residual property that will be described later.

We take the Lanczos algorithm derived in Section 3 as the first main ingredient of QMR. In its nth iteration step, ALGORITHM 3 generates

$$\mathbf{V}_{n+1} := [\mathbf{v}_1\ \mathbf{v}_2\ \cdots\ \mathbf{v}_{n+1}] \in \mathbb{C}^{N \times (n+1)} \quad \text{and} \quad \mathbf{P}_n := [\mathbf{p}_1\ \mathbf{p}_2\ \cdots\ \mathbf{p}_n] \in \mathbb{C}^{N \times n}$$

that according to (23) and (25) are connected by

$$\mathbf{P}_n = \mathbf{V}_n \mathbf{U}_n^{-1} \ , \tag{57}$$

$$\mathbf{A}\mathbf{P}_n = \mathbf{V}_{n+1} \mathbf{L}_n \ , \tag{58}$$

where

$$\mathbf{L}_n = \begin{pmatrix} \tau_1 & & \\ \gamma_2 & \ddots & \\ & \ddots & \tau_n \\ & & \gamma_{n+1} \end{pmatrix} \in \mathbb{C}^{(n+1)\times n}, \quad \mathbf{U}_n = \begin{pmatrix} 1 & \mu_2 & & \\ & 1 & \ddots & \\ & & \ddots & \mu_n \\ & & & 1 \end{pmatrix} \in \mathbb{C}^{n\times n}$$

are the leading principal $(n+1) \times n$ and $n \times n$ submatrices of $\mathbf{L}$ and $\mathbf{U}$ defined in (21). Note that $\mathbf{L}_n$ has full rank as long as no breakdown occurs.

For the QMR method, the initial residual vector $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ is used as the starting vector $\tilde{\mathbf{v}}_1$ of the Lanczos process that gives

$$\mathbf{v}_1 = \frac{1}{\gamma_1}\mathbf{r}_0 \quad \text{and} \quad \gamma_1 = \|\mathbf{r}_0\| \quad . \tag{59}$$

The second starting vector $\tilde{\mathbf{w}}_1$ may be chosen adequately, e.g. $\tilde{\mathbf{w}}_1 = \tilde{\mathbf{v}}_1$.

The second main ingredient of QMR is the definition of its iterates by the quasi-minimal residual property. In (56), the vector $\mathbf{z}_n \in \mathbb{C}^n$ is determined by making the corresponding residual vector

$$\mathbf{r}_n = \mathbf{b} - \mathbf{A}\mathbf{x}_n \tag{60}$$

small in the following sense. The setting

$$\mathbf{y}_n := \mathbf{U}_n \mathbf{z}_n$$

as well as (57) can be used to reformulate the QMR iterate (56) in terms of $\mathbf{y}_n$ instead of $\mathbf{z}_n$ giving

$$\mathbf{x}_n = \mathbf{x}_0 + \mathbf{P}_n \mathbf{y}_n \quad . \tag{61}$$

The corresponding residual vector in terms of $\mathbf{y}_n$ follows from (60) by applying the two Lanczos relations (58) and (59) resulting in

$$\mathbf{r}_n = \mathbf{V}_{n+1} \left(\gamma_1 \mathbf{e}_1^{(n+1)} - \mathbf{L}_n \mathbf{y}_n \right) \quad ,$$

where $\mathbf{e}_1^{(n+1)} = (1, 0, \ldots, 0)^T \in \mathbb{C}^{n+1}$. Rather than minimizing $\|\mathbf{r}_n\|$ that generally is an expensive task, the quasi-minimal residual property reduces costs by only minimizing the factor of the residual given in parentheses, i.e., $\mathbf{y}_n$ is the solution of the least-squares problem

$$\left\| \gamma_1 \, \mathbf{e}_1^{(n+1)} - \mathbf{L}_n \, \mathbf{y}_n \right\| \quad = \quad \min_{\mathbf{y} \in \mathbb{C}^n} \left\| \gamma_1 \, \mathbf{e}_1^{(n+1)} - \mathbf{L}_n \, \mathbf{y} \right\| \quad .$$

Since $\mathbf{L}_n$ has full rank, the solution $\mathbf{y}_n$ is uniquely determined by Lemma 4.1. Thus, the solution $\mathbf{y}_n$ is given by the two coupled recurrences of Corollary 4.1. Inserting the first recurrence relation (53) into (61) yields

$$\begin{aligned} \mathbf{x}_n &= \mathbf{x}_0 + \mathbf{P}_{n-1}\mathbf{y}_{n-1} + \mathbf{P}_n \mathbf{g}_n \\ &= \mathbf{x}_{n-1} + \mathbf{P}_n \mathbf{g}_n \\ &= \mathbf{x}_{n-1} + \mathbf{d}_n \quad , \end{aligned} \tag{62}$$

where $\mathbf{d}_n := \mathbf{P}_n \mathbf{g}_n$ is introduced. Using the second recurrence (54), the vector $\mathbf{d}_n$ is updated by

$$\begin{aligned} \mathbf{d}_n &= \theta_n \mathbf{P}_{n-1} \mathbf{g}_{n-1} + \kappa_n \mathbf{P}_n \mathbf{e}_n^{(n)} \\ &= \theta_n \mathbf{d}_{n-1} + \kappa_n \mathbf{p}_n \ , \end{aligned} \tag{63}$$

where θ_n and κ_n are defined in (50)–(52) and $\mathbf{d}_0 = \mathbf{0}$. Note that the vector $\mathbf{p}_n$ is produced by the Lanczos process. By defining $\mathbf{s}_n := \mathbf{A}\mathbf{d}_n$, the residual vector is obtained by inserting (62) into (60) giving

$$\mathbf{r}_n = \mathbf{r}_{n-1} - \mathbf{s}_n \ . \tag{64}$$

Multiplying (63) through by $\mathbf{A}$, gives a recursion for the vector $\mathbf{s}_n$ as follows

$$\mathbf{s}_n = \theta_n \mathbf{s}_{n-1} + \kappa_n \mathbf{A}\mathbf{p}_n \ , \tag{65}$$

where $\mathbf{s}_0 = \mathbf{0}$. Properly putting the four equations (62)–(65) and the three recurrences (50)–(52) of Lemma 4.1 on top of ALGORITHM 3 completes the QMR method. The resulting QMR version with one global synchronization point is depicted in ALGORITHM 4.

Remark 5.1. On the assumption that no breakdown occurs, the QMR method of ALGORITHM 4 stops if the Euclidean norm of the residual vector falls below a user-supplied tolerance. This stopping criterion adds global synchronization. The convergence check in ALGORITHM 4 is delayed such that the whole algorithm still involves just one global synchronization point. That aspect should be taken into account while implementing other stopping criteria.

Remark 5.2. The equations that are put on top of the Lanczos algorithm to build the QMR method do not require a further matrix-vector product. The vector $\mathbf{A}\mathbf{p}_n$ appearing in (65) is already known from the calculation of $\widetilde{\mathbf{v}}_{n+1}$. Consequently, a parallel implementation of ALGORITHM 4 can simultaneously compute the two matrix-vector products using the strategy outlined in Remark 3.3.

Remark 5.3. Besides the computation of two matrix-vector products, ALGORITHM 4 requires $26N + c_1$ floating-point operations (additions or multiplications) per iteration, where N is the dimension of the linear system, c_1 is a constant, and costs of the convergence check are not included. Freund and Nachtigal propose a QMR version in Algorithm 7.1 of [20], also published in [3], corresponding to ALGORITHM 4 in the sense that coupled two-term recurrences and appropriate scalings of the Lanczos vectors are used. Their algorithm consists of three global synchronization points and is formulated without residual vectors. If one adds two equations by analogy to (64) and (65) in order to recursively compute the residual vector and takes the unpreconditioned form of Algorithm 7.1, i.e., $\mathbf{M} = \mathbf{M}_1 = \mathbf{M}_2 = \mathbf{I}$, their algorithm needs $26N + c_2$ floating-point operations as well. So, the new Lanczos algorithm derived in Section 3 leads to a new QMR version that reduces the number of global synchronization points per iteration by a factor of 3 while not increasing computational costs.

Input $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{x}_0$

$\mathbf{p}_0 = \mathbf{q}_0 = \mathbf{d}_0 = \mathbf{s}_0 = \mathbf{0}$

$\lambda_1 = 1\,, \quad \kappa_0 = -1$

$\widetilde{\mathbf{w}}_1 = \widetilde{\mathbf{v}}_1 = \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$

$\gamma_1 = \|\widetilde{\mathbf{v}}_1\|\,,\ \xi_1 = \|\widetilde{\mathbf{w}}_1\|\,,\ \varrho_1 = \widetilde{\mathbf{w}}_1^T\widetilde{\mathbf{v}}_1\,,\ \varepsilon_1 = \left(A^T\widetilde{\mathbf{w}}_1\right)^T\widetilde{\mathbf{v}}_1\,,\ \mu_1 = 0\,,\ \tau_1 = \dfrac{\varepsilon_1}{\varrho_1}$

for $n = 1, 2, 3, \ldots$ **do**

$\mathbf{p}_n = \frac{1}{\gamma_n}\widetilde{\mathbf{v}}_n - \mu_n\mathbf{p}_{n-1}$

$\mathbf{q}_n = \frac{1}{\xi_n}\mathbf{A}^T\widetilde{\mathbf{w}}_n - \frac{\gamma_n\mu_n}{\xi_n}\mathbf{q}_{n-1}$

$\widetilde{\mathbf{v}}_{n+1} = \mathbf{A}\mathbf{p}_n - \frac{\tau_n}{\gamma_n}\widetilde{\mathbf{v}}_n$

$\widetilde{\mathbf{w}}_{n+1} = \mathbf{q}_n - \frac{\tau_n}{\xi_n}\widetilde{\mathbf{w}}_n$

- **if** $(||\mathbf{r}_{n-1}|| <$ tolerance$)$ **then** STOP
- $\gamma_{n+1} = \|\widetilde{\mathbf{v}}_{n+1}\|$
- $\xi_{n+1} = \|\widetilde{\mathbf{w}}_{n+1}\|$
- $\varrho_{n+1} = \widetilde{\mathbf{w}}_{n+1}^T\widetilde{\mathbf{v}}_{n+1}$
- $\varepsilon_{n+1} = \left(\mathbf{A}^T\widetilde{\mathbf{w}}_{n+1}\right)^T\widetilde{\mathbf{v}}_{n+1}$

$\mu_{n+1} = \dfrac{\gamma_n\xi_n\varrho_{n+1}}{\gamma_{n+1}\tau_n\varrho_n}$

$\tau_{n+1} = \dfrac{\varepsilon_{n+1}}{\varrho_{n+1}} - \gamma_{n+1}\mu_{n+1}$

$\theta_n = \dfrac{|\tau_n|^2(1-\lambda_n)}{\lambda_n|\tau_n|^2 + |\gamma_{n+1}|^2}$ Eq. (50)

$\kappa_n = \dfrac{-\gamma_n\overline{\tau}_n\kappa_{n-1}}{\lambda_n|\tau_n|^2 + |\gamma_{n+1}|^2}$ Eq. (51)

$\lambda_{n+1} = \dfrac{\lambda_n|\tau_n|^2}{\lambda_n|\tau_n|^2 + |\gamma_{n+1}|^2}$ Eq. (52)

$\mathbf{d}_n = \theta_n\mathbf{d}_{n-1} + \kappa_n\,\mathbf{p}_n$ Eq. (63)

$\mathbf{s}_n = \theta_n\mathbf{s}_{n-1} + \kappa_n\,\mathbf{A}\mathbf{p}_n$ Eq. (65)

$\mathbf{x}_n = \mathbf{x}_{n-1} + \mathbf{d}_n$ Eq. (62)

$\mathbf{r}_n = \mathbf{r}_{n-1} - \mathbf{s}_n$ Eq. (64)

endfor

ALGORITHM 4: QMR method with *one* global synchronization point

6 Summary

A new version of the unsymmetric Lanczos algorithm meeting the generally-known requirements of numerical stability is derived, namely, the Lanczos vectors are scaled to unit length and are generated by coupled two-term recurrences. Additionally, all inner products of a single iteration step are independent. This is a crucial feature on parallel computers with a large number of processors where the performance is almost entirely dominated by the computation of inner products that need synchronization of all processors. These favorable properties can be transfered to a broader class of algorithms, those that make use of the unsymmetric Lanczos algorithm as an underlying process. The quasi-minimal residual method is taken as an example accompanied by an elegant and new derivation.

7 Acknowledgements

The authors are gratefully indebted to R. Beucker, J. van der Linden, R. von Seggern, P. Weidner, and E.M.E. Wermuth for carefully reading earlier drafts of this paper and giving several valuable comments.

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenny, S. Ostrouchov, and D. Sorensen. *LAPACK User's Guide*. SIAM, Philadelphia, PA, second edition, 1995.

[2] Z. Bai. Error Analysis of the Lanczos Algorithm for the Nonsymmetric Eigenvalue Problem. *Mathematics of Computation*, 62(205):209–226, 1994.

[3] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, 1993.

[4] C. Brezinski, M. R. Zaglia, and H. Sadok. A Breakdown–Free Lanczos Type Algorithm for Solving Linear Systems. *Numerische Mathematik*, 63(1):29–38, 1992.

[5] H. M. Bücker. Isoefficiency Analysis of Parallel QMR-Like Iterative Methods and its Implications on Parallel Algorithm Design. Internal Report KFA–ZAM–IB–9604, Research Centre Jülich, Jülich, Germany, March 1996.

[6] J. Choi, J. J. Dongarra, and D. W. Walker. The Design of a Parallel Dense Linear Algebra Software Library: Reduction to Hessenberg, Tridiagonal, and Bidiagonal Form. Technical Report ORNL/TM–12472, Oak Ridge National Laboratory, Oak Ridge, January 1995.

[7] J. Cullum and R. A. Willoughby. A Practical Procedure for Computing Eigenvalues of Large Sparse Nonsymmetric Matrices. In J. Cullum and R. A. Willoughby,

editors, *Large Scale Eigenvalue Problems, Proceedings of the IBM Europe Institute Workshop on Large Scale Eigenvalue Problems, Oberlech, Austria, July 8–12, 1985*, number 127 in North-Holland Mathematics Studies, pages 193–240, Amsterdam, The Netherlands, 1986. North-Holland.

[8] J. Cullum and R. A. Willoughby, editors. *Large Scale Eigenvalue Problems, Proceedings of the IBM Europe Institute Workshop on Large Scale Eigenvalue Problems, Oberlech, Austria, July 8–12, 1985*, number 127 in North-Holland Mathematics Studies, Amsterdam, The Netherlands, 1986. North-Holland.

[9] J. K. Cullum and R. A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Volume I: Theory*. Birkhäuser, Boston, 1985.

[10] E. de Sturler. A Parallel Variant of GMRES(m). Reports of the Faculty of Technical Mathematics and Informatics 91–85, Delft University of Technology, Delft, The Netherlands, 1991.

[11] E. de Sturler. A Performance Model for Krylov Subspace Methods on Mesh–based Parallel Computers. Technical Report CSCS–TR–94–05, Swiss Scientific Computing Center, CH–6928 Manno, Switzerland, May 1994.

[12] E. de Sturler and H. A. van der Vorst. Reducing the Effect of Global Communication in GMRES(m) and CG on Parallel Distributed Memory Computers. Technical Report 832, University of Utrecht, Utrecht, The Netherlands, October 1993.

[13] E. F. Van de Velde. *Concurrent Scientific Computing*. Number 16 in Texts in Applied Mathematics. Springer, New–York, 1994.

[14] J. W. Demmel, M. T. Heath, and H. A. van der Vorst. Parallel Numerical Linear Algebra. In *Acta Numerica 1993*, pages 111–197. Cambridge University Press, Cambridge, 1993.

[15] J. J. Dongarra and D. W. Walker. Software Libraries for Linear Algebra Computations on High Performance Computers. *SIAM Review*, 37(2):151–180, 1995.

[16] R. W. Freund. Conjugate Gradient–Type Methods for Linear Systems with Complex Symmetric Coefficient Matrices. *SIAM Journal on Scientific and Statistical Computing*, 13(1):425–448, 1992.

[17] R. W. Freund, G. H. Golub, and N. M. Nachtigal. Iterative Solution of Linear Systems. In *Acta Numerica 1992*, pages 1–44. Cambridge University Press, Cambridge, 1992.

[18] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal. An Implementation of the Look–Ahead Lanczos Algorithm for Non–Hermitian Matrices. *SIAM Journal on Scientific Computing*, 14(1):137–158, 1993.

[19] R. W. Freund and N. M. Nachtigal. QMR: A Quasi–Minimal Residual Method for Non–Hermitian Linear Systems. *Numerische Mathematik*, 60(3):315–339, 1991.

[20] R. W. Freund and N. M. Nachtigal. An Implementation of the QMR Method Based on Coupled Two–Term Recurrences. *SIAM Journal on Scientific Computing*, 15(2):313–337, 1994.

[21] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, second edition, 1989.

[22] G. H. Golub and D. P. O'Leary. Some History of the Conjugate Gradient and Lanczos Algorithms: 1948–1976. *SIAM Review*, 31(1):50–102, 1989.

[23] A. Gupta, V. Kumar, and A. Sameh. Performance and Scalability of Preconditioned Conjugate Gradient Methods on Parallel Computers. Technical Report TR 92–64, Department of Computer Science, University of Minnesota, Minneapolis, MN – 55455, November 1992. Revised April 1994.

[24] M. H. Gutknecht. A Completed Theory of the Unsymmetric Lanczos Process and Related Algorithms, Part I. *SIAM Journal on Matrix Analysis and Applications*, 13(2):594–639, 1992.

[25] M. H. Gutknecht. A Completed Theory of the Unsymmetric Lanczos Process and Related Algorithms, Part II. *SIAM Journal on Matrix Analysis and Applications*, 15(1):15–58, 1994.

[26] T. Huckle. Low-Rank Modification of the Unsymmetric Lanczos Algorithm. *Mathematics of Computation*, 64(212):1577–1588, 1995.

[27] W. Kerner. Large–Scale Complex Eigenvalue Problems. *Journal of Computational Physics*, 85(1):1–85, 1989.

[28] S. K. Kim and A. T. Chronopoulos. An Efficient Nonsymmetric Lanczos Method on Parallel Vector Computers. *Journal of Computational and Applied Mathematics*, 42:357–374, 1992.

[29] C. Lanczos. An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators. *Journal of Research of the National Bureau of Standards*, 45(4):255–282, 1950.

[30] C. Lanczos. Solutions of Systems of Linear Equations by Minimized Iterations. *Journal of Research of the National Bureau of Standards*, 49(1):33–53, 1952.

[31] B. N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice-Hall, Englewood Cliffs, 1980.

[32] B. N. Parlett. Reduction to Tridiagonal Form and Minimal Realizations. *SIAM Journal on Matrix Analysis and Applications*, 13(2):567–593, 1992.

[33] B. N. Parlett, D. R. Taylor, and Z. A. Liu. A Look–Ahead Lanczos Algorithm for Unsymmetric Matrices. *Mathematics of Computation*, 44(169):105–124, 1985.

[34] Y. Saad. Practical Use of Polynomial Preconditionings for the Conjugate Gradient Method. *SIAM Journal on Scientific and Statistical Computing*, 6(4):865–881, 1985.

[35] D. R. Taylor. *Analysis of the Look Ahead Lanczos Algorithm for Unsymmetric Matrices*. Ph. D. dissertation, Department of Mathematics, University of California, Berkeley, CA, November 1982.

[36] Q. Ye. A Breakdown–Free Variation of the Nonsymmetric Lanczos Algorithms. *Mathematics of Computation*, 62(205):179–207, 1994.